Naive Bayes classifier for Fake News recognition

by Marco Chiloiro

Goal

The goal of the project is to implement a Multinomial Naive Bayes classifier in R and test its performances in the classification of social media posts.

Multinomial Naive Bayes Classifier (MNBC)

The probability estimation of a document d being in class *c*, with the number of terms within it equal to n_d, is computed as:

$$\hat{P}(c|d) \propto \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class *c*, while $P(c)$ is the prior probability of a document occurring in class *c*.

Assumption: independence between the features

Multinomial Naive Bayes Classifier (MNBC)

Our goal is to find the best class for the document. The best class in NB classification is the **maximum a posteriori** (MAP) class:

$$c_{\text{map}} = \text{argmax}_{c \in C} \log \hat{P}(c|d) = \text{argmax}_{c \in C} \left(\log \hat{P}(c) + \sum_{k=1}^{n_d} \log \hat{P}(t_k|c) \right)$$

- P̂(t|c) is a weight that indicates how good an indicator tk is for c
- P̂(c) is a weight that indicates the relative frequency of c. More frequent classes are more likely to be the correct class than infrequent classes.

Prior estimation

Given *Nc* the number of documents in the class *c* and *N* the total number of documents, the prior for each class *c* is:

$$\hat{P}(c) = \frac{N_c}{N}$$

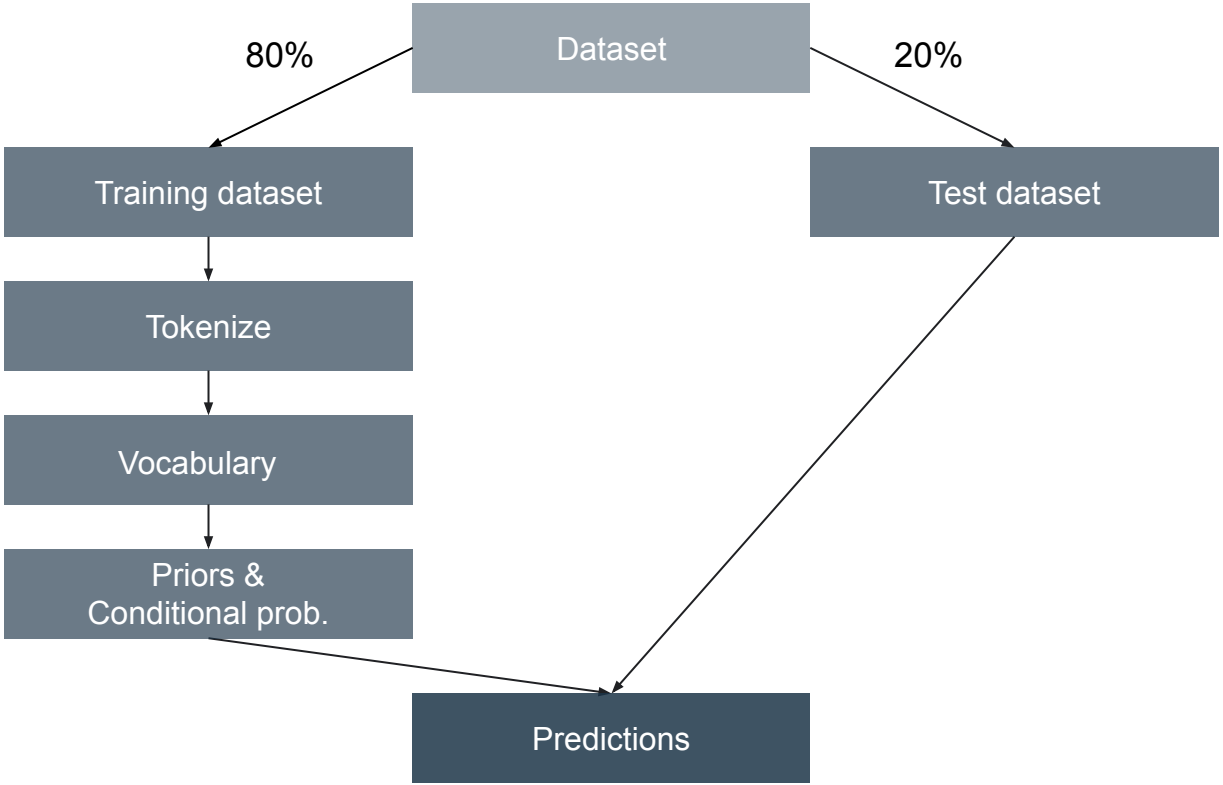
Conditional probability estimation

We estimate the conditional probability $\hat{P}(t|c)$ as the **relative frequency** of term *t* in documents belonging to class *c*, using **Laplace smoothing** to eliminate zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|}$$

where T_{ct} is the number of occurrences of *t* in training documents from class *c*, including multiple occurrences of a term in a document, and *V* is the **vocabulary**.

MNBC workflow

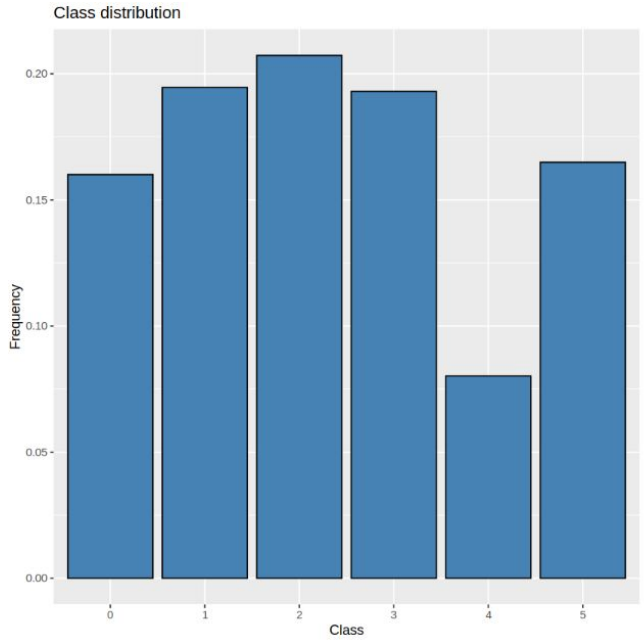


Datasets

1) **Fake News Content Detection**	**2) Fake News**
Labels: ● 5 - True ● 4 - Not-Known ● 3 - Mostly-True ● 2 - Half-True ● 1 - False ● 0 - Barely-True	Labels: ● 1 - unreliable ● 0 - reliable

Both of them can be found on Kaggle.
The dataset columns are redefined as ‘Text’ and ‘Labels’.

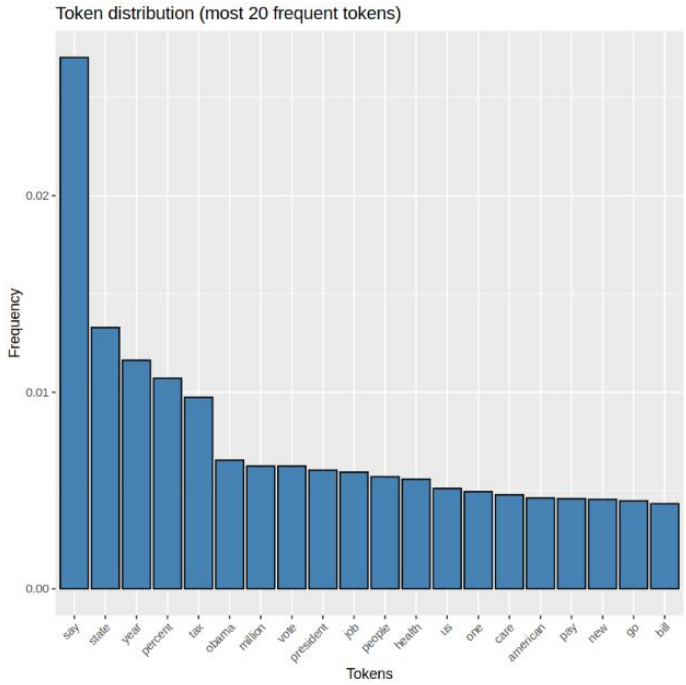
Dataset 1 - Class distribution



We can see that the classes are not balanced.

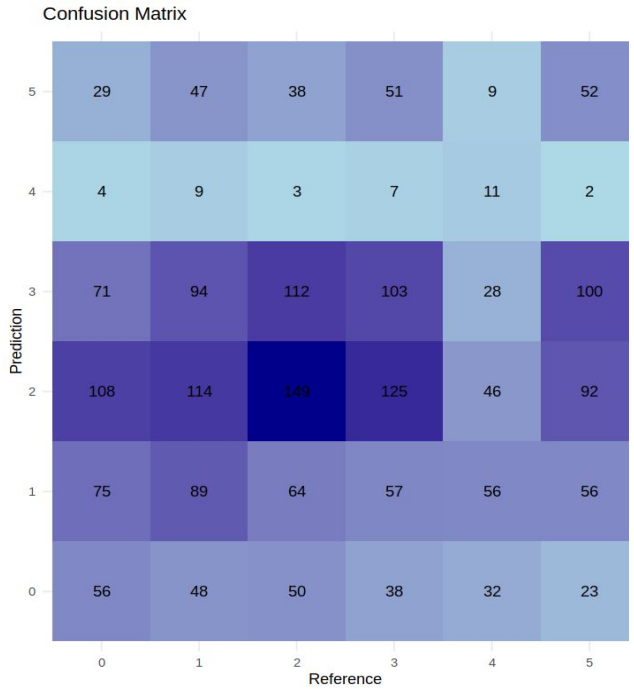
The class 4 is the less populated, while the class 2 is the more populated.

Dataset 1 - MNBC without any improvements



The 20 most frequent token, considering also the repetitions within the documents.

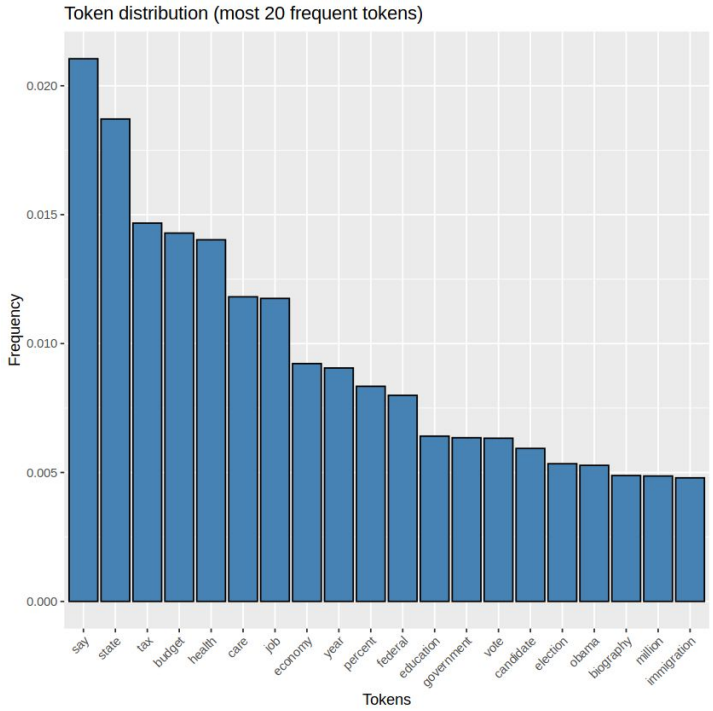
Dataset 1 - MNBC without any improvements - Results



Class	Sensitivity	Specificity
0	16.3%	88.8%
1	22.2%	81.3%
2	35.8%	70.3%
3	27.0%	75.7%
4	6.0%	98.7%
5	16.0%	89.9%

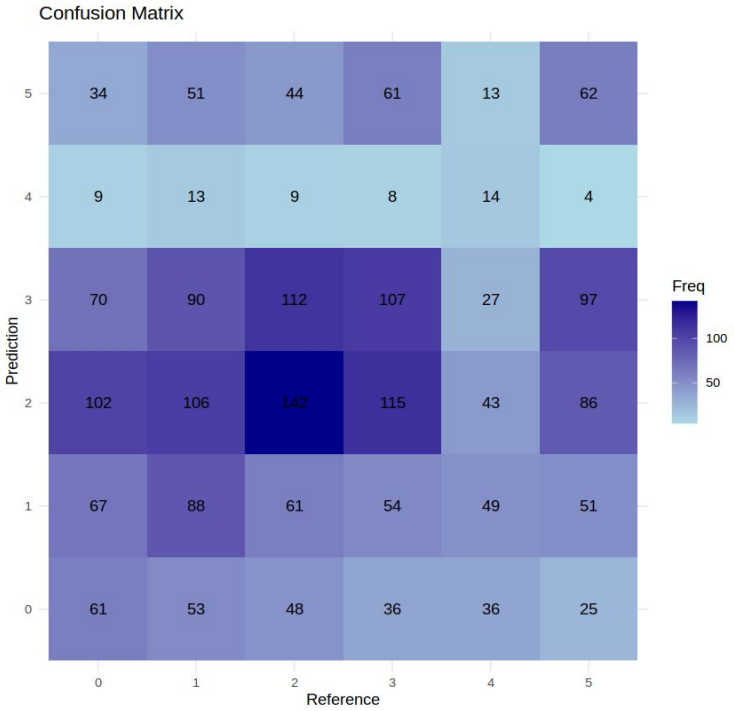
Accuracy: 22.5%

Dataset 1 - MNBC including tags



Include the column ‘Text_Tag’ within the column ‘Text’.

Dataset 1 - MNBC including tags - Results



Class	Sensitivity	Specificity
0	17.8%	88.4%
1	21.9%	82.9%
2	34.1%	72.3%
3	28.1%	76.2%
4	7.7%	97.7%
5	19.0%	88.2%

Accuracy: 23.1%

Including tags, there is a little improvement.

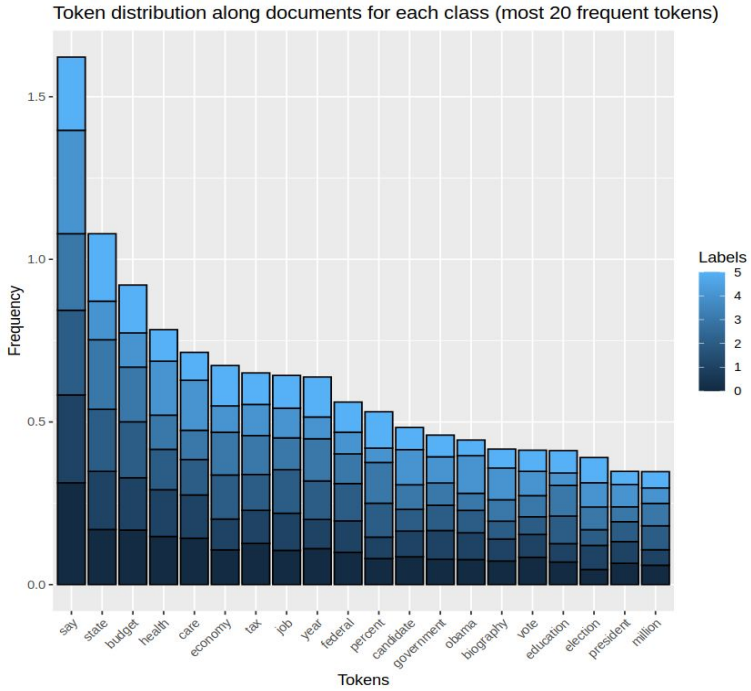
Feature selection

Feature selection involves selecting a relevant subset of features (words or terms) from the text corpus to **reduce the input dimension** and **improve the model's performance**. Practically speaking, the aim is to reduce the number of tokens within the vocabulary.

Choose a **frequency-based** feature selection: fix a threshold to remove the less frequent term for each class (without counting repetitions within the same document).

There are many ways of doing feature selection, a possible future work could be to find the best way to do that.

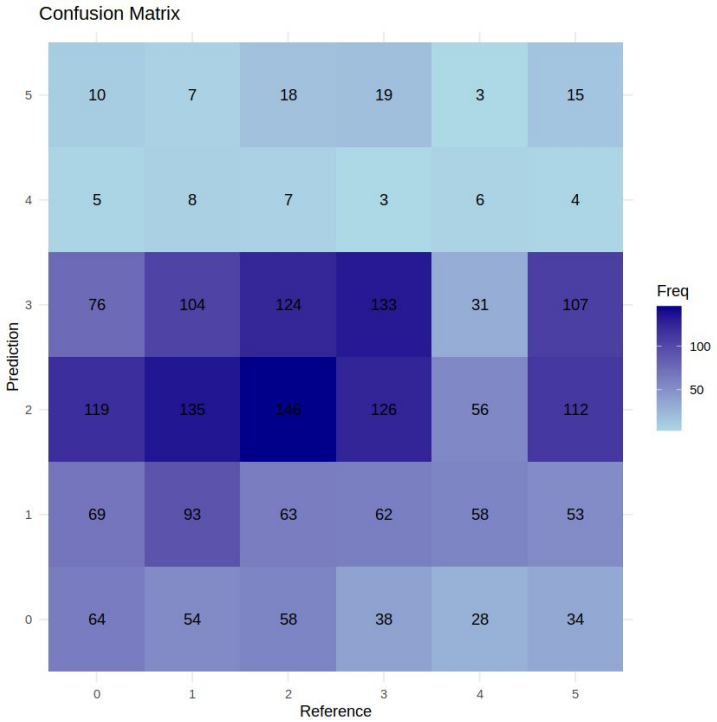
Dataset 1 - MNBC with feature selection



The idea is that the most and less frequent tokens are the less significative.
The maximum frequency is less than 50%, then an upper threshold is not necessary.

Lower threshold: 3%.

Dataset 1 - MNBC with feature selection - Results



Accuracy: 22.3%

Class	Sensitivity	Specificity
0	18.7%	87.6%
1	23.2%	81.5%
2	35.1%	66.4%
3	34.9%	73.5%
4	3.3%	98.6%
5	4.6%	96.7%

Dataset 1 - Results comparison

- Adding tags inside posts slightly increases performance
- With feature selection (threshold of 3%), the accuracy decrease from 23.1% to 22.3%
- The sensitivity for classes 2 and 3 becomes larger and for classes 4 and 5 lower, and the opposite holds for the specificity. For classes 0 and 1 the performance are almost the same as in the case without feature selection
- This behaviour suggests that there could be a correlation between the classes 5 and 3 (True and Barely-True) and the classes 4 and 2 (Not-Known and Half-True)

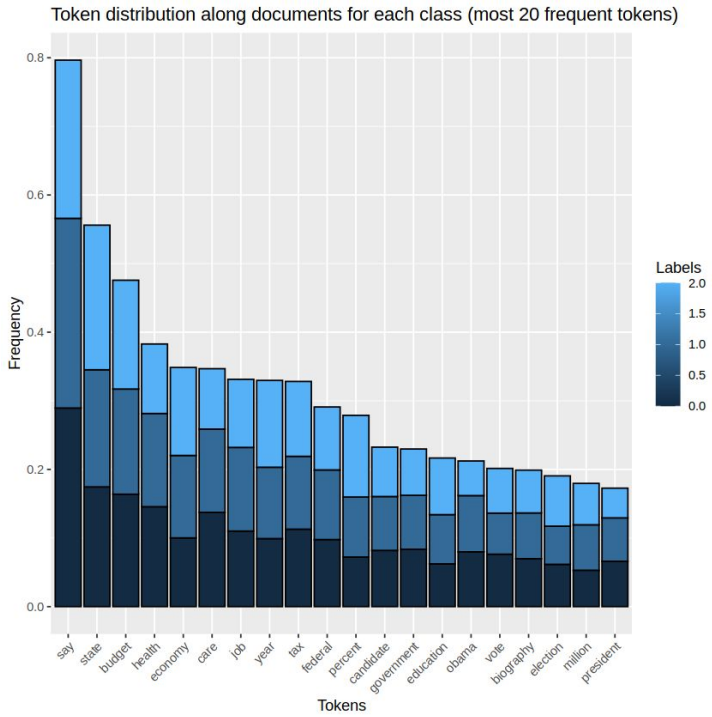
Dataset 1 - Relabelling

Considering the previous considerations, it is natural to relabel data as follows:

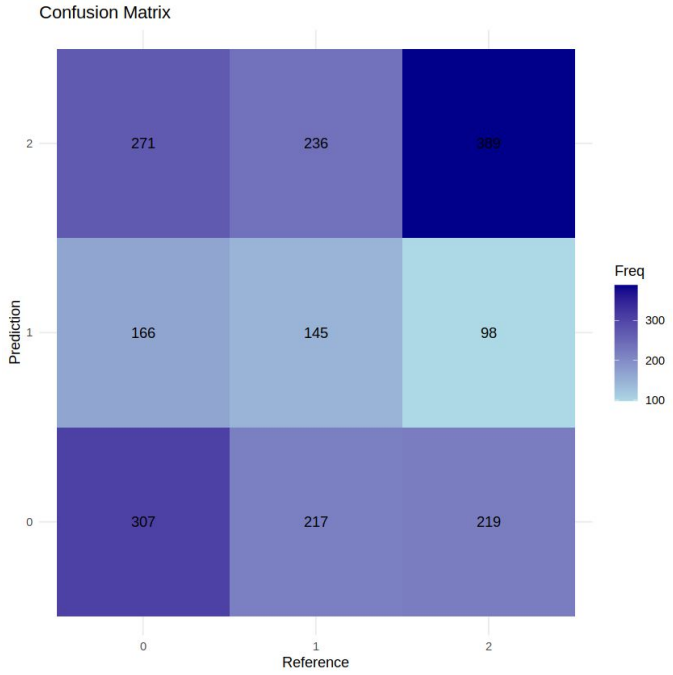
- classes 0, 1 to **class 0** (**False**)
- classes 2, 4 to **class 1** (**Maybe**)
- classes 3, 5 to **class 2** (**True**)

In this way, the classes represent a sort of ‘truth gradient’.

Dataset 1 - MNBC w/o feature selection



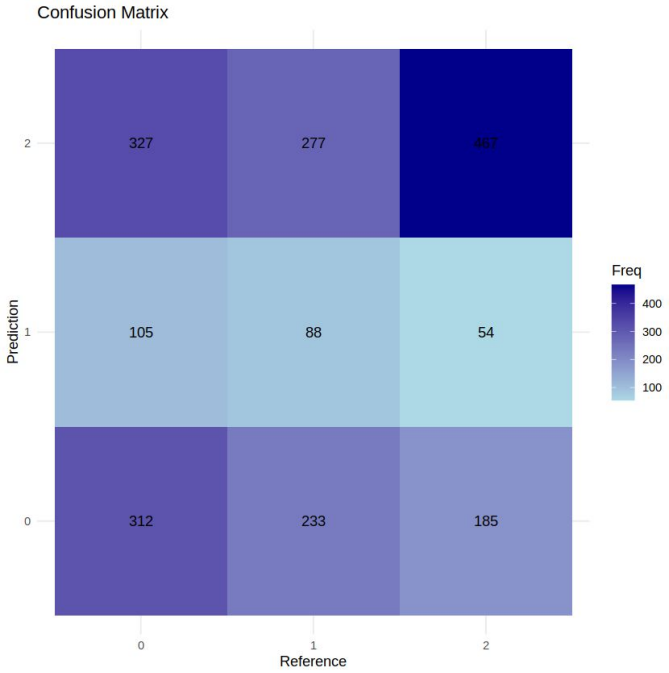
Dataset 1 - MNBC w/o feature selection - Results



Class	Sensitivity	Specificity
0	41.3%	66.6%
1	24.3%	81.8%
2	55.1%	62.2%

Accuracy: 41.1%

Dataset 1 - MNBC with feature selection - Results



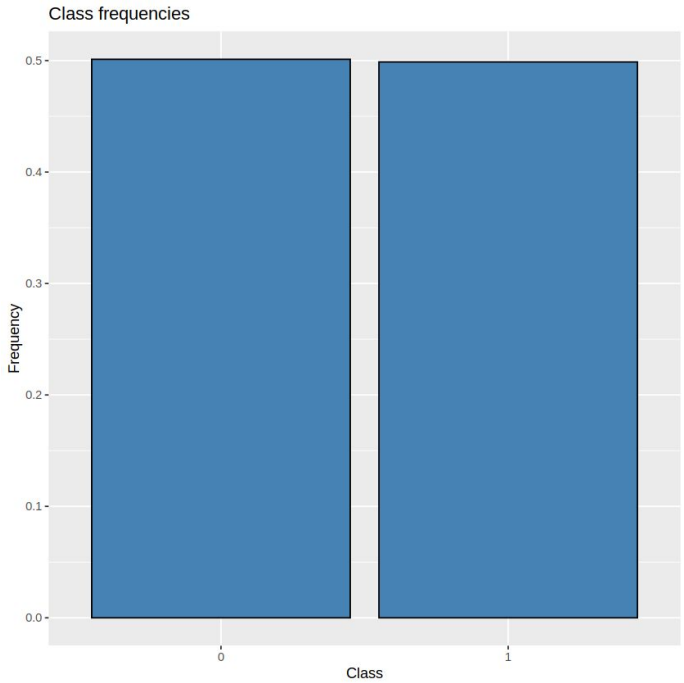
Threshold: 1%

Class	Sensitivity	Specificity
0	41.9%	67.9%
1	14.7%	89.0%
2	66.2%	55.0%

Accuracy: 42.3%

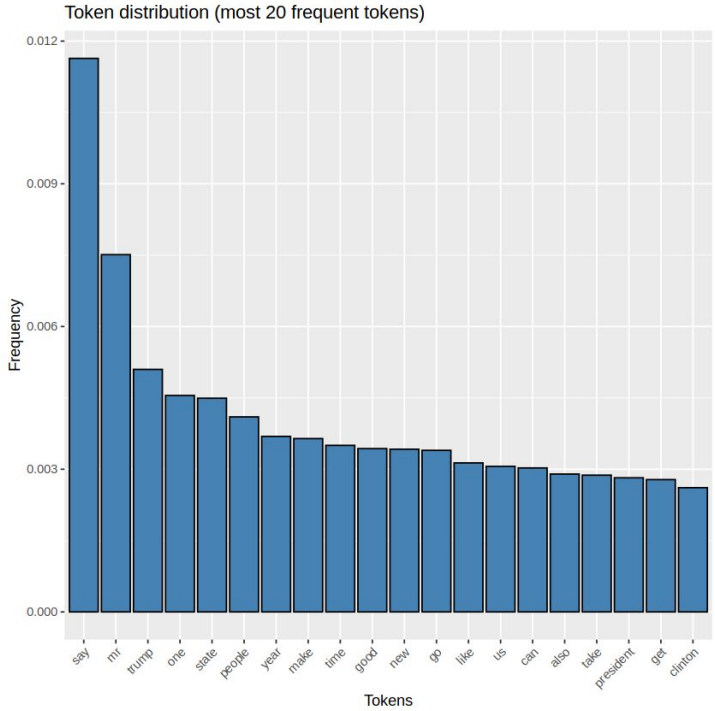
Better accuracy of about 1%

Dataset 2 - Class distribution



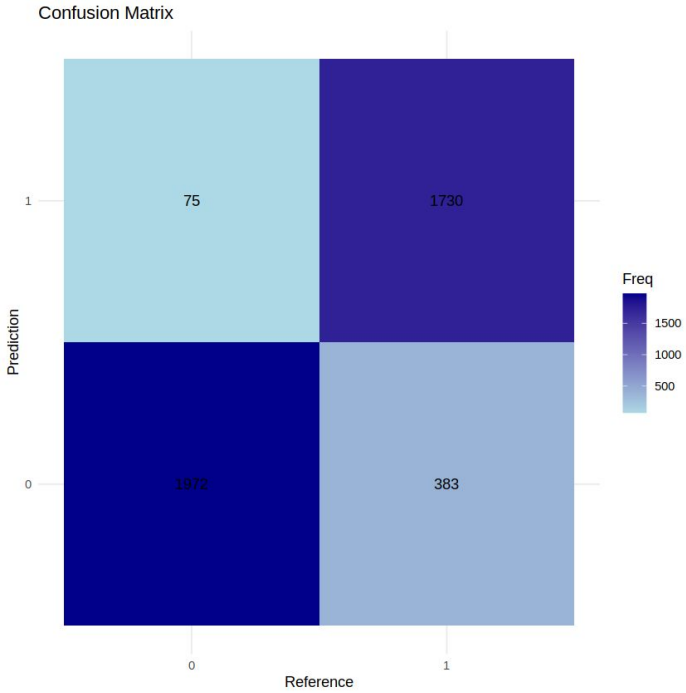
In this case, the two classes are **balanced**

Dataset 2 - MNBC without any improvements



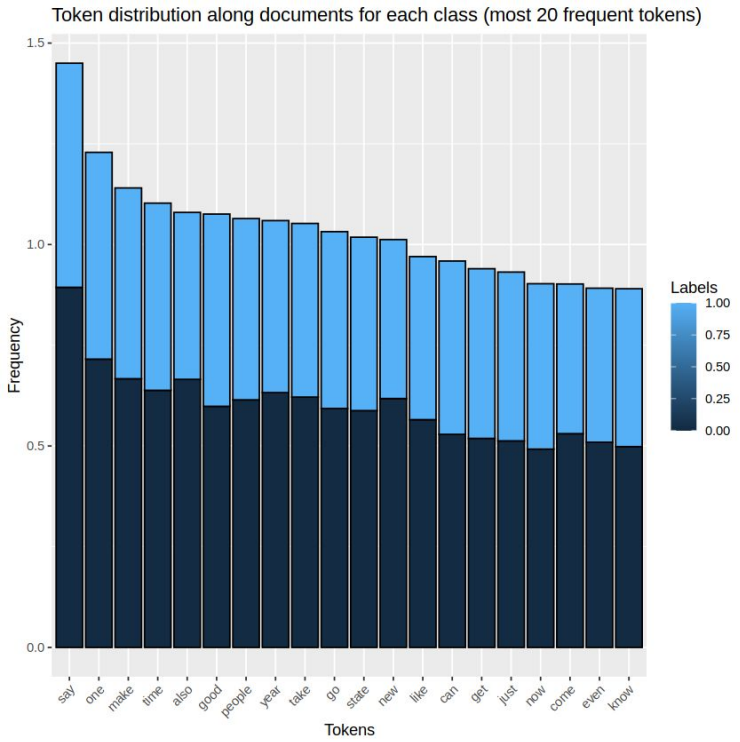
The 20 most frequent token, considering also the repetitions within the documents.

Dataset 2 - MNBC without any improvements - Results

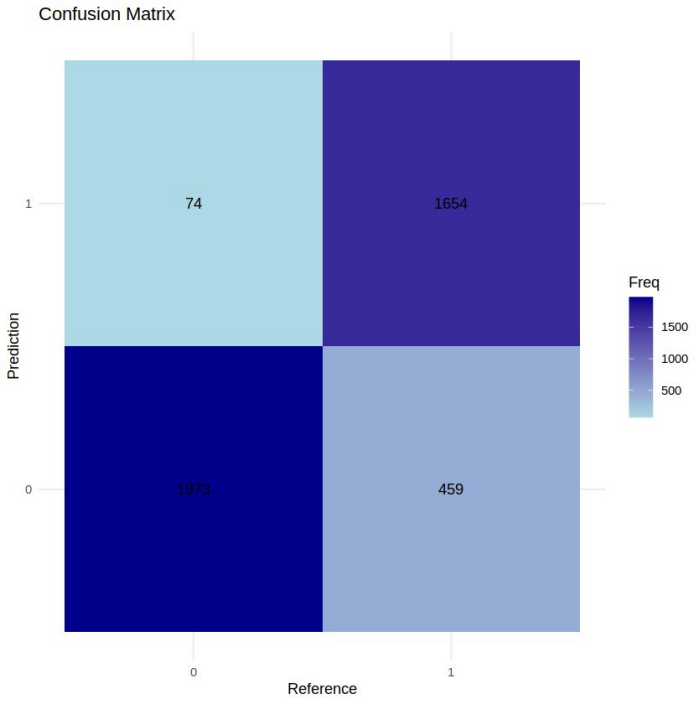


Sensitivity	Specificity	Accuracy
96.3%	81.9%	89.0%

Dataset 2 - MNBC with feature selection



Dataset 2 - MNBC with feature selection - Results



Threshold: 1%

Sensitivity	Specificity	Accuracy
96.4%	78.3%	87.2%

Conclusions

- The model works better having **more words** within each document
- The model works better considering **fewer classes**
- More the classes are **balanced** and more the model works better
- Frequency-based feature selection can work only considering **case by case**